Collaborative, Parallel Monte Carlo Tree Search for Autonomous Electricity Demand Management

Fatemeh Golpayegani, Ivana Dusparic, Siobhán Clarke
School of Computer Science and Statistics
Trinity College Dublin
Dublin, Ireland

***Abstract*— Balancing electricity supply and consumption is critical for the stable performance of an electricity Grid. Demand Side Management (DSM) refers to shifting consumers' energy usage to off-peaks as much as possible to avoid more electricity demand than available supply during peak times. Artificial intelligent planning algorithms have been applied to enabling electric devices to reschedule their operation to off-peaks. One such algorithm is Monte Carlo Tree Search (MCTS), which takes advantage of tree search and random sampling on decision space in order to find an optimal domain decision.**

In particular for DSM, MCTS has been used to control both smart meters and also electrical devices. In both applications, MCTS acts as a centralized consumption planner choosing the optimum approach for all devices in the space. Centralized computation thread limits these approaches in terms of flexibility and scalability. As applied in domains outside DSM, an alternative, decentralized MCTS algorithm, called Parallel MCTS (P-MCTS), allows every agent to run its independent MCTS thread to have its own solution.

In this paper, first we studied the feasibility of applying P-MCTS in DSM to make demand planning more flexible and scalable. P-MCTS has been evaluated by running two different scenarios one with 6 electrical vehicles (EVs) and the other with 90 EVs which roughly results in 30% peak load shifting for P-MCTS. To improve the results of P-MCTS, we propose a new decentralized collaborative approach, called Collaborative P-MCTS (CP-MCTS), which exploits and extends P-MCTS to enable each electrical device to actively affect the planning process but also to improve the final decision using collective knowledge obtained during the collaboration. Additionally, in comparison with P-MCTS, CP-MCTS obtained better results, including more peak load shifting and smoother load curve due to 30% lower Peak to Average Ratio (PAR).

Keywords— Monte Carlo Tree Search, Collaboration, Demand Side Management.

I. MOTIVATION

Demand Side Management (DSM) is a set of techniques run by utility companies to optimize consumers' electricity consumption [1]. An important goal for DSM is to make the most of current energy capacity, avoiding renewable energy waste and unnecessary energy use at peak times and reducing the need to increase capacity. A range of approaches to achieve these goals are at varying stages of maturity including energy efficiency, fuel substitution, demand- response and residential/ commercial load management [2] and [3].

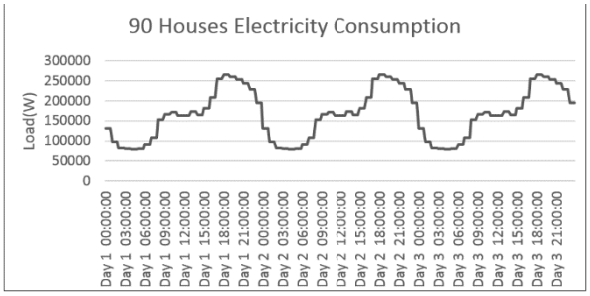

Fig. 1. Electricity consumption for 90 houses from Ireland smart meter trial

Residential/commercial load management includes consumption reduction and/or load shifting [4]. Residential energy consumption is not evenly distributed during a day. To illustrate this, Fig. 1 shows the electricity consumption in a community of 90 houses over a period of 3 days, which shows morning peaks and evening peaks relative to varying daily activities of the inhabitants. With increasing penetration of electrical vehicles (EVs), household load will be almost doubled [5]. Consequently, higher peaks lead to more electricity production requirements. DSM aims to smooth the load by shifting high peak noncritical consumption to low peak times or intelligently distribute energy consumption over time to decrease the peak-to-average ratio (PAR) in load demand.

Different approaches have been applied to address residential load management. These approaches can be categorized into two broad categories, direct and indirect approaches. Direct approaches, where a supplier is engaged and is responsible for controlling households (e.g., [6] and [7]). Indirect approaches enable devices to use energy in a smarter and more efficient way using artificial intelligence methods. The second category includes approaches which consider dynamic pricing, scheduling, prediction and learning to control and shift the load. To facilitate load balancing, prediction based approaches determine if a suitable (e.g., cheaper energy, less load demand) time slot exists in the near future so the consumption can be delayed [8] and [9]. Also, learning approaches enable electrical devices to learn what is the best action to do in each time slot [10]. The indirect category itself can be divided into centralized smart methods, and decentralized smart methods. Centralized approaches, involve a controller (i.e., a coordinator which has access to each household data and the current load, generates a schedule

978-3-901882-70-8

for other households) which generates a coordinated consumption schedule for households based on the current demand load, price and households priority status [11] and [12]. Scheduling can be categorized as a centralized approach in which computational complexity and communication requirements influence system's scalability [13]. Although Centralized approaches have reported effective results, they have been applied mostly for small scale scenarios. Furthermore, households misbehaving or failure on running the given schedule, would result in an overhead on system to reschedule the whole plan considering the new situation [14].

Decentralized approaches engage smart elements (e.g., smart meters, smart devices) to determine their consumption plan. To illustrate this, [15] is a smart decentralized method where agents that control electrical devices try to learn from their previous actions and forecasted load for a better action selection in future. While [16],[17] and [18] are decentralized approaches, they still use a smart meter in each house to control consumption and defer the overload to off peaks. The smart meters decide and act as a house level controller. In Multi-Agent Systems (MAS), devices are controlled by agents which have goals and can take a range of actions to reach their goals. Agents act separately or together and decide about their actions. In [15], each agent looks for the best action in each stage based on what it has learned or what is predicted. Although decentralized learning based approaches are promising and have reported effective results, agents still need to spend time on exploring and learning, which affects their performance at the very early stages.

To address these issues, Monte Carlo Tree Search (MCTS) is a promising approaches that has been recently used in DSM and MAS. MCTS is a search method which tries to find the optimal decision by forming a search tree based on the random samples of the decision space. Since 2006, MCTS has remarkable achievements in computer games and AI planning. Apart from advances in computer games, it also has been used in optimization problems, planning and learning [15]. MCTS has been used in DSM, as a centralized electricity consumption planner. It has been run by smart meters, electrical devices to find the best electricity consumption pattern considering electricity price and electricity base load [11], [19] and [20]. These approaches used MCTS as a central controller, and the devices and smart meters are simple players that run their provided consumption schedule without any manipulation.

MCTS approaches that have been used in DSM have certain limitations. Firstly, they used a central controller/coordinator object which makes the approach unscalable due to a massive computation load that should be carried by that single thread. Secondly, since MCTS is a central coordinator of the system, the other players (agents) do not have autonomy to determine their own decisions and should wait for dictated plan. On the other hand, a Parallel variant of MCTS (P-MCTS), which is the distributed implementation of MCTS, has been used in games domain. It allows players to have their own control thread separated and independent from other players without any communication or coordination. The final result of P-MCTS probably will be an optimal partial solution for each player but it might not be necessarily an optimal solution for the whole system.

To address these problems, in this paper we aim for two goals; first we study the feasibility of P-MCTS in DSM and then we propose the main contribution of this paper as a new decentralized multi-agent method called Collaborative Parallel Monte Carlo Tree Search (CP-MCTS). CP-MCTS is applied in a DSM scenario -including a number of houses with a number of electrical devices and electrical vehicles (agents) - to address a decentralized load management issue using a combination of artificial intelligent planning, collaboration and collective decision making. More specifically, in this approach agents are determinant members of the whole load management process rather than a simple player which executes the final schedule. Moreover, CP-MCTS enables agents to collaborate to make more coordinated and optimal decisions autonomously.

The rest of this paper is organized as follows. Section II discusses most related literature. Section III presents an overview of Monte Carlo Tree Search and specifically Parallel Monte Carlo Tree Search. Section IV introduces CP-MCTS. Section V presents experiment design. Section VI describes empirical evaluation and the results. Finally, conclusion and future works are discussed in Section VII.

II. BACKGROUND

One of the first applications of MCTS in DSM was presented in [11]. In this paper, MCTS was applied to find the best consumers' participation rate, which is a consumers' consumption strategy based on dynamic pricing. The optimum solution is the one with lower PAR. In this approach, while it builds the search tree, a fraction of consumers are assigned a particular participation rate in each level of the MCTS tree, until the tree reaches its leaf. It means that all the consumers are assigned to a participation rate. Results are evaluated by the most commonly used performance metrics such as PAR and area under PAR (AUP). This approach is applied to consumers' electricity meter level rather than device level and the control of each device is not automated.

Heuristic Multi Agent MCTS (MA-MCTS) [19], is a variant of MCTS, which is extended using a heuristic to be applicable for a multi-agent system context. The heuristic mentioned in this work plays a vital role, as it determines the amount of charge each EV needs to be charged in each time slot. Each EV has four different options for energy consumption in each time step (i.e., 0, 1/3, 2/3 and 1). If one EV is assigned to 1/3, this means it is allowed to be ON for the 1/3 time step. These values are assigned to each EV by a function (heuristic). This function simply distributes the energy consumption over the time by knowing about the total demand of all EVs and total number of time slots. Therefore, EVs which are controlled by agents are not considered autonomous and they are not able to decide about their charging plan while they play whatever is dictated to them by MCTS. In addition, there is no data privacy for the EVs, while they have to reveal their daily journey plan and their state of charge to the main algorithm. This work runs a single thread MCTS algorithm for all the EVs while considering a vector of 6 EVs, suggesting it might not be scalable to larger implementation.

In [20], a DSM scenario including four devices in a house is modelled using the two players game approach and Max^n - a variant of MCTS. In the two players approach, players play

against each other while in Max^n, although each agent tries to maximize its payoffs (e.g., rewards), it considers the other agents actions as well. During the first approach, four households are divided into two groups which run MCTS separately and they are not connected to each other at any stage. Within each group, each device plays against the other one to find the best time slot (e.g., lowest electricity price and enough time to finish their task) to turn on. In this approach, MCTS manages dependencies between the devices (e.g., dryer should be turned on after washing machine) and finds the best solution for a small group of electrical devises. The problem is that each group is isolated from other groups and is just looking for the best solution for itself rather than the best solution for the whole system. In the Max^n approach, all the devices are trying to maximize their own electricity share while also considering other devices statuses. Therefore, devices use MCTS to find the best solution but the difference is that these devices are added to the tree one by one during the specified time constraint. Although this method is better than the two players one while it is considering the other devices, due to fixed time constraint assumption, it does not perform very well in medium scale or large scale scenarios. If it implements a bigger number of devices, there would be less time for simulation devoted for each device. This means the results would not be as accurate as the small scale scenario since MCTS accuracy is highly dependent to the number of simulation. Besides, in this approach, a higher priority is assumed for the devices that are added to the tree first.

III. Monte Carlo Tree Search

A. MCTS/ Upper Confidence Bound for Trees

MCTS is a search method which tries to find the optimal decision by forming a search tree based on the random samples of the decision space. MCTS has four main phases: Selection, Expansion, Simulation and Back-Propagation. In the selection phase, the player starts traversing the tree from its root, until it reaches a leaf node. Then MCTS moves to its second phase to add a new node to the tree, which expands the tree with this new leaf node. Afterwards, the simulation will be started from that new leaf node and it will be ended when it reaches termination criteria. The last phase is Back-Propagation, which propagates the results of a simulated game backwards. Although there are several variants of MCTS, these four phases are fundamental common phases [21].

MCTS behaviour and its outcome specifically depend on the way the search tree has been built, and building the tree depends on the algorithm which selects the nodes (moves) during the selection phase. Upper Confidence Bound for Tree (UCT) is a popular variant of MCTS which addresses these issues and it is selected to be the basis of this paper as well. UCT tries to balance both exploration and exploitation, by considering the moves that have been explored so far and new moves to check if they can improve the results [22]. Therefore, UCT considers the move that maximizes the result of equation (1).

$$UCT = \overline{X_j} + 2C\sqrt{\frac{2\ln n}{n_j}} \qquad (1)$$

where n is the number of times, parent node has been visited, n_j is the number of times, child node has been visited, C is a constant and $\overline{X_j}$ is within [0, 1]. When more than one child exist with the same maximum value, one of them randomly will be selected [23].

B. Parallel Monte Carlo Tree Search

Parallel Monte Carlo Tree Search (P-MCTS) is a variant of MCTS that runs parallel threads, at least in one of the four MCTS phases. P-MCTS has been categorized into three different approaches based on the specific MCTS phase/phases that parallelization occurs [24].

- The first approach is Leaf Parallelization [22], in which the simulation step of MCTS is parallelized. When a new node is added to the tree, multiple threads start the simulation independently and in parallel. Therefore, instead of waiting for one thread to simulate all the possible actions, one by one, several threads do the simulation.
- The second approach is Tree Parallelization [24]. In this approach there is a shared tree which can be accessed and modified by any parallel thread. Mutexes are used to lock the tree at certain period of times (e.g., when more than one thread tries to modify the tree) to prevent data corruption.
- The third type of parallelization which is exploited in this paper is Root Parallelization [22]. The Root Parallel approach is basically a parallel implementation of UCT. This approach runs in two modes, Master mode and Slave mode. During the Slave Mode each player runs MCTS separately and independently from other players on its own thread. Therefore, each player forms its own MCTS tree, which might be different from others' trees (e.g., Fig2), and never shares any information with other players. As it is shown in Fig. 2, at each time step each thread runs the MCTS and does the simulation. Master Mode starts when the simulation time is finished. During this process, the master adds the partial results provided by each player and assumes it as the final results.

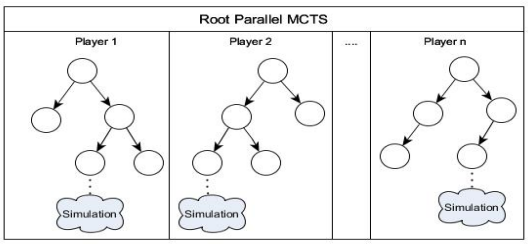

Fig. 2. Root Parallel MCTS Mechanism

IV. Collaborative Parallel Monte Carlo Tree Search

In the Leaf Parallel approach, multiple threads are used just during the simulation while the rest of the approach is the same as non-parallel MCTS, and as a result there is one MCTS tree which stores all the information and solution in itself. In the Tree Parallel approach, although there are multiple threads that can run MCTS in parallel, there is still one MCTS tree

(knowledge base) that is shared and there is a central control thread to avoid data corruption. On the other hand, Root Parallel is the only approach that can be implemented in a decentralized manner. Each player (agent) has its own independent thread and never communicates or shares information with any other. Therefore, this approach has the highest potential to be implemented in a decentralized manner.

Root Parallel approach runs in Master/Slave mode: in Slave mode, players run the MCTS independently in parallel and when the time is up, the master tries to put the partial results together to create a final MCTS tree during Master mode. The main limitation of this approach is the central decision maker (Master), which makes the final decision based on partial results independently from the players. In other words, in the presence of any conflicts or constraint violation, the Master may decide about the suitable action. This limitation is addressed in our CP-MCTS, by implementing collaboration between agents which are contributing to a conflict or constraint violation.

A. CP-MCTS Algorithm

In this section, we propose a new collaborative approach based on P-MCTS, which is called Collaborative Parallel Monte Carlo Tree Search (CP-MCTS). CP-MCTS, as it is shown in Fig. 3, includes two stages: Parallel Stage and Collaborative Stage. During the Parallel Stage, each agent runs MCTS in parallel and independently from others, creating a decision tree with possible actions. When it reaches the termination criteria, it checks its action with other agents and if there is any constraint violation, it enters the Collaborative Stage. In the Collaborative Stage, the agent which finds the violation, searches for other agents which contributed to this situation and they start a negotiation and take the final decision collectively. These two stages are discussed in following sections.

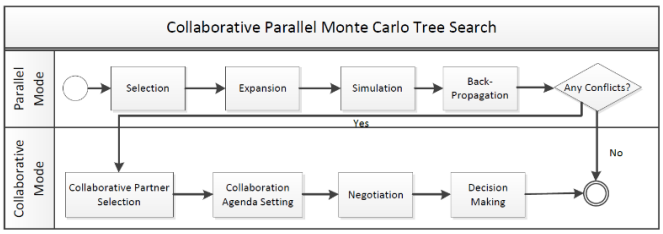

Fig. 3. CP-MCTS Process

1) Parallel Stage: The Parallel Stage consists of five steps. Each agent creates its own tree by running four steps of Selection, Expansion, Simulation and Back-Propagation. Afterwards, agents have to make the final decision. In Root Parallel MCTS, the Master always concludes and makes the final decision based on the partial results provided by each agent. However, in CP-MCTS, there is no Master role to coordinate the partial results and generate the final result. The agents themselves, have to decide collectively about the final result. CP-MCTS modifies P-MCTS as follows:

a) Selection: In this step, each agent starts to travers the tree from the root to the node that is still expandable. It selects one of the children of that node and goes to the next phase.

b) Expansion: In this step, the child selected from the previous step is added to the tree.

c) Simulation: The agent starts simulation from the added node, while it has not reached the termination condition.

d) Back-Propagation: In this step, all the route from the added node to the root is traversed backward and all the rewards of each node in this route will be updated by new values.

e) Any Conflicts*:* In this step, each agent checks what will happen, if it takes its next action, considering the systems constraints/goals and other agents' next action. If its action violates any constraint or does not contribute to the satisfactory of the overall goal, it starts the Collaborative Stage, otherwise each agent will take its next action.

```
Collaborative_Stage:

CollaborationPartnerSelection ()
1  For i ← 0 to collaborationcommunity.size ()
2    Collaborationcommunity[i] ← Agent.ID;
3  Return (Collaborationcommunity []);
CollaborationAgendaSetting ()
1  CollaborationGoal ← Community.goal;
2  CollaborationPlan ← Community.Plan;
3  for i ←1 to Collaborationcommunity.size ()
4    Agent[i].setCollaborationPriority ();
Negotiation (Threshold)
1    for i ← 0 to Collaborationcommunity.size ()
2      for j ← 0 to Collaborationcommunity.size ()
3        if (Agent[i].Priority- Agent[j].Priority > Threshold)
4            N[i][j] ← 1;
5        else
6            N[i][j] ← 0;
7    Return N [ ][ ];
DecisionMaking (N[ ][ ])
1    for i ← 0 to Collaborationcommunity.size ()
2      for j ← 0 to Collaborationcommunity.size ()
3            Decision[i]+=N[i][j];
4  While (!CollaborationGoal)
5            i ← 0;
6            Index ← Min (Decision [i]);
7            Agent[index]..Action ← OFF;
8            i++;
9            if (!AnyConflict()) Break;
```

Fig. 4. Collaborative Stage Pseudocode

2) Collaborative Stage: Agents enter this stage if they face any conflicts or constraint violation. This stage consists of four steps.

a) Collaboration Partner Selection: During this step, the agent which has identified the conflict, searches for other agents which have contributed to this situation. It sends them notification and asks them to start a collaboration. As it is shown in Fig. 4, the initiator agent searches the whole community for the agents have contributed to the conflict. Finally, the collaboration community is formed.

b) Collaboration Agenda Setting: The collaboration agenda is defined based on three main elements: collaboration

community goal, collaboration community plan and agents individual priorities. In this step, as it is shown in Fig. 4, the collaboration community's goal and plan are the same as community's goal and plan, though in other cases, they may be different. Agents determine their priorities based on their current state and ultimate goal. While agents are determinig their priority, a predefined threshold helps them to scale their priority. (e.g., if the battery charge of an EV is below 10%, then the priority is high, therefore this 10% is defined as a threshold).

c) Negotiation: In this step, as it is outlined in the algorithm in Fig. 4, each agent compares its current priority with other agents in the collaboration community. In each comparison, the agent with higher priority receives a negotiation score. This process continues until all agents get the chance to negotiate with each other. At the end of this step, the agents fill out the result matrix, based on the scores they have got against each other. When EVs have the same priorities, they try to find different priorities by changing the threshold. If it could not help them to distinguish their priorities, then a random generator helps to choose one of them.

d) Decision Making: During this step, agents make the final decision based on the previously discussed priorities. Therefore, in this stage some of the agents, which their plans are not urgent, postpone their actions.

V. EXPERIMENT DESIGN

The goal of our experimental evaluation is twofold. First, we evaluate P-MCTS and CP-MCTS in a small scale scenario that was proposed in [17]. In this scenario, we compare the results with the other variant of MCTS that has been implemented in DSM with the same scenario, to show that CP-MCTS does not have any negative effects on the results. Afterwards, we evaluate our new approach, CP-MCTS, by implementing a bigger scale scenario. These scenarios are discussed in following sections.

A. DSM Scenarios

In our DSM scenarios, there is a community of houses with some electrical devices, including electrical vehicles (EVs) based on Nissan Leaf characteristics [25]. All the devices in the house are used regularly, which create transformer's base load. EVs have different electricity consumption pattern due to their different daily plan -including distance to work and departure/arrival time. These scenarios are implemented in a power distribution simulator called GridLAB-D [26].

In a Greedy charging approach, which is considered as a baseline in our evaluation, EVs start charging as soon as they get back home. While most of the EVs get back from work during the evening peak, if they start to charge in the same time period, they will double the peak load. Therefore, two main goals are considered in these scenarios:

- Noncritical load shifting during peak time.
- Providing enough battery charge for each EV to complete its daily journey.

1) Scenario I: In this scenario, we study the feasibility of the P-MCTS by considering the same assumptions of the previously implemented MCTS approach in DSM [17]. This scenario considers a community of 6 houses, with one EV in each house. EVs leave home around 6 am and they get back home around 6 pm. These EVs need to be charged while they are home, to have enough battery charge to accomplish their next journey. Each EV can change its charging state (i.e., ON or OFF) every 30 minutes.

2) Scenario II: To evaluate CP-MCTS in this scenario, we consider a bigger community with 90 houses and one EV in each with different daily plan. The rest of the assumptions are the same as the first scenario.

B. DSM Scenario Parameter Setting

1) Actions and Reward Function: In this simulation, MCTS tree structure depends on the number of actions (i.e., which is equal to maximum number of edges in each node) and the reward function. The possible actions for each EV at each time step is ON or OFF. Therefore, the MCTS tree in this simulation has a maximum of two edges in each node. The right edge simulates ON action and the left edge simulates OFF action.

The reward function, which directly affects the tree structure, considers three main elements including base load, Sate of Charge (SoC) and number of hours left until next journey. Therefore, each EV has to evaluate its actions based on these criteria, when it is building its MCTS tree.

2) Termination Criteria: Since the number of actions during the simulation is always constant, the simulation does not terminate naturally in each round. Therefore, we use time constraint t to stop the simulation. t is assumed constant for all the agents to make a fair situation for all the EVs. Based on the DSM scenarios discussed earlier, the EVs should decide their next action at the end of each time slot (e.g., every 30 minutes).

C. CP-MCTS applied in DSM

This section explains the CP-MCTS implementation for DSM scenario.

1) Agents in Parallel Stage: When EVs reach home, they charge as soon as possible. Therefore, they start running the parallel stage of CP-MCTS. Each agent builds the search tree independently until it reaches its termination condition. The agents' next action is either turn ON or OFF. The EVs which their next selected action is ON, search for any potential constraint violation. To do so, they check the current load and the other agents' next action. In presence of any constraint violation, they start the Collaborative Stage.

2) Agents in Collaborative Stage: The first step in the Collaborative Stage is to form a community of collaborative agents. This community is formed from the EVs which their next selected action is ON. The first EV which notices a potential constraint violation, searches and invites others to form collaboration community. After community formation, the EVs set up a collaboration agenda including collaboration

goal/s and plan/s. Also, they set up their priorities as outlined in the algorithm in Fig. 4. In this scenario they set the collaboration goal to reach a charging pattern without any constraint violation (e.g., lower transformer load) in each timestep. Accordingly, the plan is to rearrange the charging plan to meet the goal. Also, each EV sets a charging priority based on its SoC, and the amount of time left to its departure. Then, the negotiation is started. In this step, each EV checks its priority with other EVs, and fills in a *m*×*m* negotiation matrix, where *m* is the number of EVs as shown in Fig. 5. For example, consider that EV1 is negotiating with EV2, while it has a higher priority than EV2. Therefore, the N(1, 2) equals to 1 and the N(2, 1) equals to 0 (e.g., Fig. 5). The higher priority determines the winner of negotiation. The final step is to make the final decision about the charging pattern. Finally, the EVs with higher secondary priorities (e.g., the priority acquired after negotiation in matrix N) will be selected one by one, until the constraint is not reached.

$$N = \begin{array}{c} \\ EV1 \\ EV2 \\ \vdots \\ EVm \end{array} \begin{array}{c} \begin{array}{cccc} EV1 & EV2 & \cdots & EVm \end{array} \\ \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \cdots \\ 1 & 0 & \cdots & 0 \end{bmatrix} \end{array}$$

Fig. 5. Negotiation Matrix Sample

VI. RESULTS

The results presented in this section are evaluated using the performance criteria introduced in first sub-section including PAR, the transformer load curve and the EVs' SoC. In sub-section B, we compare our results with the results provided in [19] based on Scenario I, and in sub-section C, we evaluate our results based on Scenario II.

A. Performance Criteria

1) Peak-to-Average Ratio (PAR): PAR measures the performance of the system based on the demand load. In other words, it measures how the new approach affects the load demand, specifically the peaks. It is defined in equation (2).

$$PAR = \frac{Max\ Load}{\sum_{t=0}^{n} load_t} * n \qquad (2)$$

This equation calculates the maximum load divided by average load; where $Max\ Load$ is the maximum load in a day, *n* is the number of time slot in each day and $load_t$ is the recorded load for time slot *t*. Lower PAR means smoother transformer load curve, which implies load demand is distributed better over the time [27].

2) Transformer Load: The Transformer Load plots the load distribution over time. Therefore, it is used to demonstrate that the load distribution is aligned with the first defined goal (i.e., noncritical load shifting during the peak time) of this experiment.

3) State of Charge(SoC): SoC plots the battery charge percentage for each EV before and after charging in each day. This plot is reported to show that the second goal (i.e., providing enough battery charge for each EV to complete its daily journey*)* is fulfilled.

B. Evaluation based on Scenario I

In this section, first we evaluate feasibility of P-MCTS by comparing it with the Greedy approach and the PAR results that has been presented in MA-MCTS [19]. Then, we compare CP-MCTS results versus P-MCTS and MA-MCTS. To make the comparison possible, all the assumption in [19] are assumed in Scenario I (e.g., Including number of EVs and their daily plan).

PAR results obtained from four different approaches are depicted in Fig. 6. It is shown that the results achieved by P-MCTS, which are fluctuating between 1 and 1.8, are better than the results obtained from Greedy approach, which is fluctuating between 2.5 and 3.7. This implies that P-MCTS creates a smoother load curve by shifting noncritical load to off-peaks.

The results obtained from P-MCTS in Scenario I, can be compared to MA-MCTS [19] in different aspects. Firstly, MA-MCTS is a centralized approach that uses a coordinator which determines the final charging plan is not violating the pre-defined constraints. Also, MCTS in this approach acts as a central computation thread, while EVs have no authority on running the MCTS or changing their charging plan. On the other hand, P-MCTS is a decentralized approach, which is executed by the agents while each agent can implement its internal reward function based on its data and functionality. Therefore, there is no central coordinator to control, determine and align EVs' (agents') actions towards the final goal. Thus, each EV finds its best solution advised by its search tree considering the whole constraint. Furthermore, it is more scalable while the computation load is distributed amongst the agents. Although this approach is much more flexible and scalable than the MA-MCTS, it also has certain limitations. Since EVs are not communicating and coordinating their results, having the best partial results do not result in the best final decision.

The results obtained from CP-MCTS show that the limitations of P-MCTS are tackled. As it was discussed earlier, agents coordinate their actions towards their goals and their community goal through collaboration. Therefore, if there was any potential constraint violation, they collaborate to find a solution. As it is shown in Fig. 6, CP-MCTS has achieved 30% less PAR compare to P-MCTS and also it delivers an improvement in compare to MA-MCTS.

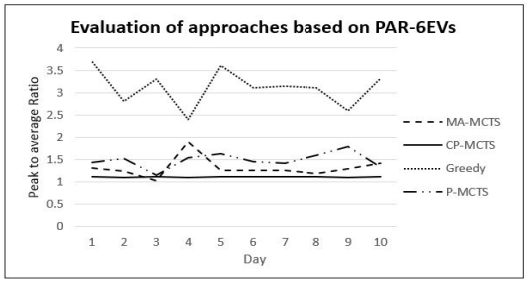

Fig. 6. Peak to Average Ratio Comparison over 10 days

C. *Evaluation based on Scenario II*

To evaluate the scalability of CP-MCTS, we run experiments for a larger community with 90 EVs. Also, to evaluate CP-MCTS's performance and flexibility, we run the experiment in three different situations. To do so, Scenario II has been run with three different distances to work. The first scenario (i.e., 30 miles) implements the situation that EVs can get enough charge if they postpone their charging to off-peak hours. The second scenario (i.e., 45 miles) implements the situation that agents need more hours than off-peaks to be charged enough. Finally, the third scenario (i.e., 70 miles) considers the situation that EVs should be charging the whole time that they are at home to get enough charge for next day.

1) Scenario II- 30 miles: We report the results obtained from Transformer Load, PAR and SoC for this scenario.

a) Transformer Load results: The transformer load results from implementation of three different approaches including CP-MCTS, P-MCTS and Greedy is reported in Fig. 7. In Greedy approach, EVs start charging as soon as they get back home without any consideration about the current demand load. As it is shown, they increase the load exactly during the peak times and end up charging around low demand time. Therefore, this approach not only produces higher peaks on the demand curve but also it does not use the energy capacity during the off-peak times. On the other hand, in P-MCTS, each agent has shifted its energy consumption to off-peaks by running MCTS in parallel with others. Therefore, running this approach results in shifting massive amount of consumption to off-peaks and using available energy capacity. During the peak times, it is still increasing the peak that is due to lack of coordination between the agents' actions. On the other hand, CP-MCTS results show that it makes the most of the available energy capacity by shifting the load to off-peaks. Also, it creates lower peaks compare to P-MCTS during the peak times using collaboration. Therefore, based on the transformer load in Fig. 7, CP-MCTS has obtained the best results by shifting the load to off-peaks and avoiding noncritical energy consumption during the peak times.

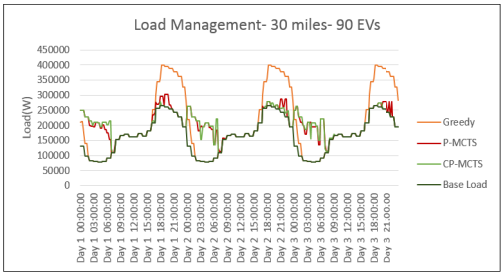

Fig. 7. Load Management using different approaches- 30 miles

b) PAR results: The results obtained from P-MCTS and CP-MCTS are also compared based on PAR in Table 1. As it is stated in this table, CP-MCTS achieved around 28% better result which implies lower peaks and smoother load curve than P-MCTS.

TABLE I. PEAK-TO-AVERAGE RATIO BASED ON SCENARIO II

Scenario II – 30 miles		Day 1	Day 2	Day 3
PAR	*CP-MCTS*	1.11	1.13	1.08
	P-MCTS	1.53	1.56	1.48
	Improvement	28%	28%	27%

c) SoC results: EVs decide whether to charge or not to charge based on their current state (e.g., its current SoC and amount of time left to next journey). Regardless of their decisions, it is important to be charged enough for the next journey. EV1's SoC is plotted out in Fig. 8, as a sample to show that it is charged enough every day to complete its next day journey. As it is shown in this figure, EV1 charges almost 100% in the first day, but it is not fully charged for the second day while it still has enough charge to continue for the next day. Finally, in third day, it charges almost 100% again.

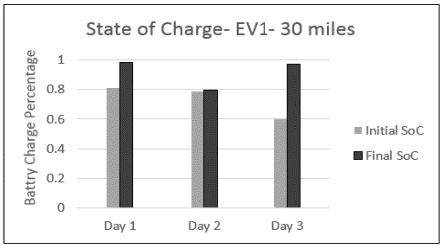

Fig. 8. Percentage of battery charge in each day before and after charging

2) Scenario II- 45 miles: The results obtained from implementing this scenario are shown in Fig. 9, where EVs need more hours than off-peaks to be charged. The point is that, although some of the EVs charge during the peak times, they avoid charging all together at the same time slot via collaboration. As it is shown in Fig. 9, CP-MCTS obtained better results including lower peaks during peak times and smoother load curve, compare to P-MCTS and Greedy approach.

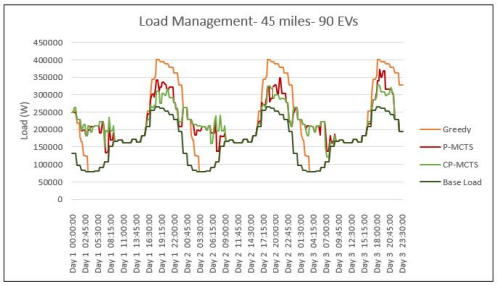

Fig. 9. Load Management using different approaches- 45 miles

3) Scenario II- 70 miles: By increasing the distance to 70 miles, EVs need to be charged almost all the hours they spend at home. The aim of this scenario was showing that CP-MCTS and P-MCTS do not defer the charging process to the off-peak but also they consider the overall situation to make the best decision. As it is shown in Fig. 10, EVs charge all the time they are at home. Therefore, the transformer loads obtained from the three different approaches are more or less the same.

Fig. 10. Load Management using different approaches-70 miles distance

VII. Conclusion and Future Works

In CP-MCTS, we used the concepts of P-MCTS and UCT, to propose a new decentralized collaborative approach. CP-MCTS enables agents with parallel threads to collaborate and find an optimal solution autonomously. This approach applied in DSM to address the residential load management to achieve two main goals including smoother load demand (lower PAR) and providing enough battery charge for each EV. With respect to the literature, our approach has 3 main advantages: (i) it provides an intelligent decision making/conflict resolution approach for P-MCTS through collaboration, (ii) it is more scalable than the previous MCTS approaches used in DSM due to its parallel computation, (iii) it is more dynamic, that any kind of household can be easily inserted to this approach.

The results presented in this work can be extended in several directions. First, the proposed Collaborative Stage can be extended to include collaboration coordinator role that can facilitate collaboration in large scale systems by defining most related agent to avoid massive communication between agents. Second, it is interesting to extend negotiation step to include different types of negotiation to obtain further improvement on decision making process. Third, the DSM scenario can be extended to include other electrical devices (e.g., heating/cooling devices, water heater) with more variation on electricity consumption.

Acknowledgment

This work was supported by Science Foundation Ireland grant 10/CE/I1855 to Lero – the Irish Software Engineering Research Centre (www.lero.ie).

References

[1] G. Masters, "Renewable and efficient electric power systems," Hoboken, NJ: Wiley, 2004.

[2] B. Ramanathan and V. Vittal, “A framework for evaluation of advanced direct load control with minimum disruption,” IEEE Trans., Power Syst., vol. 23, no. 4, pp. 1681–1688, 2008.

[3] C. Gellings and J. Chamberlin, "Demand Side Management: Concepts and Methods," 2nd ed. Tulsa, OK: PennWell Books, 1993.

[4] Ontario Clean Air Alliance, “Energy Efficiency Strategy,” 2011.

[5] A. Mohsenian-Rad, V. Wong, J. Jatskevich, R. Schober and A. Leon-Garcia, “Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid,” IEEE Trans., Smart Grid, vol. 1, no. 3, pp. 320–331, 2010.

[6] N. Ruiz, I. Cobelo and J. Oyarzabal, “A direct load control model for virtual power plant management,” IEEE Trans., Power Syst., vol. 24, no. 2, pp. 959-966, 2009.

[7] A. Gomes, C. H. Antunes and A. G. Martins, “A Multiple Objective Approach to Direct Load Control Using an Interactive Evolutionary Algorithm,” IEEE Trans., Power Syst. vol. 22, no. 3, pp. 1004-1011, 2007.

[8] A. Faruqui and S. Sergici, "Household response to dynamic pricing of electricity: a survey of 15 experiments," Journal of Regulatory Economics vol. 38, no. 2, pp. 193-225, 2010.

[9] A. Marinescu, I. Dusparic, and C. Harris, S. Clarke and V. Cahill, “A Dynamic Forecasting Method for Small Scale Residential Electrical Demand,” proceeding of the International Joint Conference on Neural Networks. 2014.

[10] A. Taylor, E. Galván-López, S. Clarke and V. Cahill, "Accelerating Learning in Multi-Objective Systems through Transfer Learning," proceeding of a Special Session on Learning and Optimization in Multi-Criteria Dynamic and Uncertain Environments at the International Joint Conference on Neural Network. 2014.

[11] T. Wijaya and T. Papaioannou, “Effective consumption scheduling for demand-side management in the smart grid using non-uniform participation rate,” proceeding of Sustainable Internet and ICT for Sustainability (SustainIT), IEEE, 2013.

[12] T. Logenthiran, D. Srinivasan, and T. Shun, “Demand side management in smart grid using heuristic optimization,” IEEE Trans., Smart Grid, vol. 3, no. 3, pp. 1244–1252, 2012.

[13] A. Taylor, et al. “Self-Organising Algorithms for Residential Demand Response,” proceeding of Technologies for Sustainability (SusTech), IEEE, 2014.

[14] D. Hammerstrom, et al., “Pacific Northwest GridWise ™ Testbed Demonstration Projects Part I . Olympic Peninsula Project,” 2007.

[15] I. Dusparic and C. Harris, A. Marinescu, V. Cahill and S. Clarke, "Multi-agent residential demand response based on load forecasting." proceeding of Technologies for Sustainability (SusTech), pp. 90-96. IEEE, 2013.

[16] S. Ramchurn, P. Vytelingum, A. Rogers and N. Jennings, “Agent-Based Control for Decentralised Demand Side Management in the Smart Grid,” proceeding of the 10th International Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, pp. 5-12, 2011.

[17] F. Schweppe, B. Daryanian, and R. Tabors, “Algorithms for a spot price responding residential load controller,” IEEE Trans., Power Syst., vol. 4, no. 2, 1989.

[18] P. Vytelingum, T. Voice, S. Ramchurn, A. Rogers, and N. Jennings, “Agent-based Micro-Storage Management for the Smart Grid,” proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, pp. 39-46, 2010.

[19] E. Galván-López, R. Li, C. Patsakis, S. Clarke, and V. Cahill, “Heuristic-Based Multi-Agent Monte Carlo Tree Search,” proceedings of the Fifth International Conference on Information, Intelligence, Systems and Applications, IEEE, 2014.

[20] E. Galván-López and C. Harris, L. Trujillo, K. Rodriguez-Vazquez, S. Clarke and V. Cahill “Autonomous Demand-Side Management System Based on Monte Carlo Tree Search,” proceedings of International Energy Conference (EnergyCon), pp. 1325-1332, 2014.

[21] C. Browne, et al. "A survey of monte carlo tree search methods," IEEE Trans., Computational Intelligence and AI in Games, vol. 4, no. 1, pp. 1-43, 2012.

[22] T. Cazenave and N. Jouandeau, “On the parallelization of UCT,” proceedings of the Computer Games Workshop, pp. 93-101, 2007.

[23] L. Kocsis and C. Szepesvári, “Bandit based monte-carlo planning,” proceedings of the 15th European Conference on Machine Learning, pp. 282-293, 2006.

[24] G. Chaslot, M. Winands, and H. van Den Herik, “Parallel monte-carlo tree search,” Computer Games, Springer Berlin Heidelberg, pp. 60-71, 2008.

[25] U.S. EPA Fuel Economy Information. Nissan leaf, 2014.

[26] U.S. Department of Energy at Pacific Northwest National Laboratory. Gridlab-d, 2014.